NORTH-HOLLAND

Local Expression Languages for Probabilistic Dependence

Bruce D'Ambrosio
Department of Computer Science, Oregon State University, Corvallis, Oregon

ABSTRACT

A Bayesian belief net is a factored representation for a joint probability distribution over a set of variables. This factoring is made possible by the conditional independence relationships among variables made evident in the sparseness of the graphical level of the net. There is, however, another source of factoring available which cannot be directly represented in this graphical structure. This source is the microstructure within an individual marginal or conditional distribution. We present a representation capable of making this intradistribution structure explicit, and an extension to the SPI algorithm capable of utilizing this structural information to improve the efficiency of inference. We discuss the expressivity of the local expression language, and present early experimental results showing the efficacy of the approach.

1. INTRODUCTION

A Bayesian belief net [16] is a compact, localized representation of a probabilistic model. The key to its locality is that, given a graphical structure representing the dependencies (and, implicitly, conditional independencies) among a set of variables, the joint probability distribution over that set can be completely described by specifying the appropriate set of marginal and conditional distributions over the variables involved. When the graph is sparse, this will involve a much smaller set of numbers than the full joint distribution. Equally important, the graphical structure can be used to guide processing to find efficient ways to evaluate queries against the model. For more details, see [16, 19, 5, 20, 14, 13, 15].

All is not as rosy as it might seem, though. The graphical level is not capable of representing all interesting structural information which might

Address correspondence to Bruce D'Ambrosio, 303 Dearborn Hall, Oregon State University, Corvallis, OR 97331-3202. E-mail: dambrosio@cs.orst.edu.

Received September 1993; accepted August 1994.

International Journal of Approximate Reasoning 1995; 13:61–81

655 Avenue of the Americas, New York, NY 10010
0888-613X/95/$9.50
SSDI 0888-613X(94)00038-5

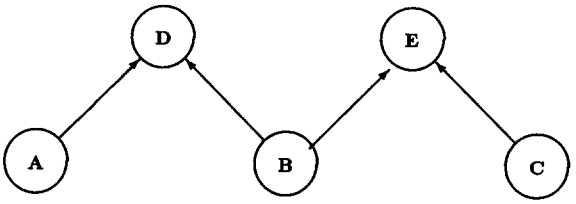

Figure 1. Noisy OR sample net.

simplify representation or inference. The only mechanism available for describing antecedent interactions in typical general-purpose Bayesian belief-net inference implementations is the full conditional distribution across all antecedents. However, a number of restricted interaction models have been identified which have lower space and time complexity than the full conditional. The noisy OR [16, 17, 9, 12, 1], for example (Figure 1), can be used to model independent causes of an event, and inference complexity is linear in both space and time in the number of antecedents for many inferences. Similarly, contingencies [21], as well as the value-dependent independence exploited in similarity nets [11], are inefficiently represented using a full conditional. In this paper we present an extension to the standard Bayesian belief-net representation which is capable of explicitly capturing much of this lower-level structural detail, and which permits use of these structures within arbitrary belief nets. We further present extensions to a factoring-based [15] inference algorithm in the SPI (symbolic probabilistic inference) family which capture both the space and the time advantages of these structures. In the remainder of this paper we first present an extension to the Bayesian belief-net representation, and show how it can be used to capture the noisy OR and various asymmetries. We then present a very brief overview of SPI in its current form. While the representation is independent of SPI and could be used within any standard algorithm for inference in Bayesian belief nets, we find SPI the most natural framework within which to discuss the issues involved in inference with the extended representation. We then discuss the actual inference-algorithm extensions we have made, and compare our local expression language with other recent efforts to generalize Bayesian nets. We close with some remaining questions.

2. LOCAL EXPRESSION LANGUAGES FOR PROBABILISTIC KNOWLEDGE

In this section we present an extension to the standard representation for belief nets. This extended expression language is useful for compact

representation of a number of canonical interaction models among antecedents. We demonstrate its use in capturing the noisy OR as well as several other prototypical interaction structures. Later sections will show how one inference family, SPI, can be extended to make efficient use of the information in the extended representation.

The local expression (that is, the expression which describes numerically the dependence of the values a variable can take on the values of its antecedents) in a Bayesian net is simple: it is either a marginal or a conditional probability distribution. While this representation is complete (that is, is capable of expressing any coherent probability model), it suffers from both space and time complexity limitations: both the space and time (for inference) required are exponential in the number of antecedents. However, computation of child probabilities using the noisy OR interaction model is linear in the number of (independent) antecedents in both space and time. When evidence is available on child variables, computation of the posterior probability of parents is exponential in the number of pieces of positive evidence, but linear in the number of pieces of negative evidence.

Heckerman [9] has developed an algorithm, called Quickscore, which provides this efficiency for two-level bipartite graphs. However, I am unaware of any implemented system other than the one reported here which can efficiently incorporate a noisy OR within an arbitrary Bayesian net. If the interaction between the effects of A and B on D in the net shown in Figure 1 can be modeled as a noisy OR interaction, then we might write the following expression for the dependence of D on A and B, following Pearl [16]:

$$P(D = t) = 1 - [1 - c_A(D)][1 - c_B(D)],$$

$$P(D = f) = [1 - c_A(D)][1 - c_B(D)],$$

where $c_A(D)$ is the probability that D is true given that A is true and B is false.[1] We use c rather than p to emphasize that these are not standard conditional probabilities. We will use a slightly more compact notation. We can define

$$c'_A(D) = 1 - c_A(D),$$

where

$$c'_A(D) = \begin{cases} 1 - c_A(D), & A = t, \\ 1, & A = f, \end{cases}$$

[1] In the Mycin terminology, $P(D = t)$ is the *probabilistic sum* of $c_A(D)$ and $c_B(D)$.

That is, whereas $c_A(D)$ is a single number $P(D = t \mid A = t, B = f)$, $c'_A(D)$ is a pair of numbers expressing the dependence of $P(D)$ on A. Now we can reexpress the above as

$$P(D = t) = 1 - c'_A(D) * c'_B(D),$$
$$P(D = f) = c'_A(D) * c'_B(D).$$

This notation is intuitively appealing. It is compact (linear in the number of antecedents), captures the structure of the interaction, and, as Heckerman et al. have shown [10], can be manually manipulated to perform efficient inference. However, it is not sufficiently formal to permit automated inference. We define a formal syntax for our expression language as follows:

$$\begin{aligned}
\textit{exp} &\rightarrow \textit{term} \mid (+ \textit{term term-set}) \\
&\rightarrow (- \textit{term term-set}) \mid \\
&\rightarrow (* \textit{term term-set}). \\
\textit{term} &\rightarrow \textit{exp} \mid \textit{distribution}. \\
\textit{term-set} &\rightarrow \textit{term} \mid \textit{term term-set}. \\
\textit{distribution} &\rightarrow \textit{name}_{\textit{dimensions}}. \\
\textit{dimensions} &\rightarrow \textit{conditioned} \mid \textit{conditioned} \text{ ``}\mid\text{''} \textit{conditioning}. \\
\textit{conditioned} &\rightarrow \textit{variable-name}_{\textit{domain}} \\
&\rightarrow \textit{variable-name}_{\textit{domain}}\ \textit{conditioned}. \\
\textit{conditioning} &\rightarrow \textit{variable-name}_{\textit{domain}} \\
&\rightarrow \textit{variable-name}_{\textit{domain}} \mid \textit{conditioning}. \\
\textit{domain} &\rightarrow \text{`` ''} \mid \textit{value} \mid \textit{value-set}. \\
\textit{value-set} &\rightarrow \textit{value} \mid \textit{value}, \textit{value-set}.
\end{aligned}$$

Notice that every term eventually must reduce to one or more *distributions*. Each distribution, in turn, is defined over some rectangular subspace of the Cartesian product of domains of its conditioned and conditioning variables. Examining the simple noisy OR example provided earlier, we discover that the informal representation obscured the fact that the two instances of $c'_A(D)$ are in fact operating over disjoint domains. In the remainder of this paper we will use the following notation to specify expressions and distributions[2]:

$$\textit{exp}(D) = 1_{D_t} - c'_{D_t \mid A_{t,f}} * c'_{D_t \mid B_{t,f}} + c'_{D_f \mid A_{t,f}} * c'_{D_f \mid B_{t,f}}.$$

[2] We ignore the actual numeric values in the distribution, since they are not germane to the discussion. Our actual syntax uses prefix notation; however, for readability we will use infix notation in the remainder of the paper.

Note that in this representation there are two instances of $c'_A(D)$. While the numeric distributions are identical, the domains over which they are defined differ.

Having specified a syntax and shown that a noisy OR can be expressed in this syntax, we will next examine the semantics for the language, and whether or not these semantics match those standardly attributed to the noisy OR structural model. Expression semantics are quite simple to specify:

> An expression is equivalent to the distribution obtained by evaluating it using the standard rules of algebra for each possible combination of antecedent values.

Performing this evaluation symbolically for our simple example yields

		D	
A	B	t	f
t	t	$1 - [1 - c_D(A)]*[1 - c_D(B)]$	$[1 - c_D(A)]*[1 - c_D(B)]$
t	f	$1 - [1 - c_D(A)]*(1)$	$[1 - c_D(A)]*(1)$
f	t	$1 - (1)*[1 - c_D(B)]$	$(1)*[1 - c_D(B)]$
f	f	$1 - (1)(1)$	$(1)(1)$

This is, in fact, exactly the standard semantics attributed to noisy OR. The next question one might ask is whether all local expressions allowed by the syntax are semantically well formed, that is, correspond to coherent implicit full conditional distributions. Unfortunately, the answer is clearly no. In fact, coherent expressions can be composed of components which individually seem to violate coherence constraints (even as plausible constraints as the requirement that values lie in the range [0, 1]). We shall see examples of this below, in considering the use of local expressions to represent additive value models. A student has developed an algorithm for coherence-checking a local expression; a technical report on this topic is in preparation.

We next consider the use of the local expression language to represent several other commonly occurring intradistribution structures.

2.1. Asymmetries

It has been pointed out that probabilistic relationships are often asymmetric [8]. In the example presented by Geiger and Heckerman, a variable *Badge* (B) is dependent on a second variable, *Hypothesis* (H), and also dependent on a third variable, *Gender* (G), only when *Hypothesis* is either "worker" or "executive" [*Hypothesis* takes four values, "worker" (w),

"executive" (e), "visitor" (v), and "spy" (s)]. We can capture this structure as follows:

$$exp(Badge) = P_{B_{t,f}|H_{s,v}} + P_{B_{t,f}|H_{w,e},G_{m,f}}.$$

Another form of asymmetry occurs when a variable has a contingent existence.[3] For example, W might only be defined when $X = t$, and might depend on the values of Y and Z [21]:

$$exp(W) = P_{W_{t,f}|X_tY_{t,f}Z_{t,f}} + P_{|X_f}.$$

The second term in the above expression may be unexpected. Remember, however, that in a normal Bayesian network we must be able to recover the joint distribution across all variables by forming the product of all distributions. The extension for local expressions is that we must be able to recover the joint distribution by forming the product of all local expressions. One easy way to ensure this is to require that every local expression be defined for every intantiation of its parent set. For each such instantiation, the local expression for a variable must then describe the distribution of the mass for that instantiation. The second term in the above expression records that the mass associated with certain instantiations of the parents (all those in which $X = f$) is not assigned to any value in the domain of the variable W. Nonetheless, this mass must be accounted for so that this expression can be properly combined with others to recover the full joint distribution.

Dagum and Galper [4] recently noted the utility of additive decomposition of conditional distributions. Their model is easily expressed in our local expression language:

$$exp(Y) = a_1 * p_{Y_{t,f}|X1_{t,f}} + a_2 * p_{Y|X2_{t,f}}.$$

Two notes about this example. First, the local expression language does not restrict the domains in the above example to $\{t, f\}$; we use that domain merely for concreteness. Second, as Dagum and Galper point out, the decomposition can be into conditioning subsets rather than individual variables.

Finally, decision models can be represented as Bayesian nets, as noted by Cooper [3]. Value structures are often factorable. Consider, for exam-

[3] Proper evaluation of queries on Bayesian nets including contingent variables goes beyond the scope of this paper; for now we will merely demonstrate the expressivity of the representation.

ple, the classic drilling example [18]. The overall utility is the value of the oil recovered, minus the cost of drilling and the cost of testing:

$$P(V) = P_{V_{t,f}|Oil\text{-}recovered_{soaking,wet,dry}} - P_{V_{t,f}|Drill_{yes,no}} - P_{V_{t,f}|Test_{yes,no}}.$$

In this case it is interesting to look at sample numeric values in the various component distributions[4]:

```
(distribution Dvalue/or          ((value t    f))
  ((oil-recovered soaking wet dry))
  ((              soaking)            1.0  0.0)
  ((                      wet)        0.6  0.4)
  ((                          dry)    0.26  .74))

(distribution Dvalue/drill ((value t   f))
      ((drill y n))
      ((      y)                  .2  -.2)
      ((        n)               0.0  0.0))

(distribution Dvalue/test ((value t   f))
      ((test y n))
      ((     y)                 .03  -.03)
      ((       n)               .0    .0))
```

There seems to be something very strange in the above, namely a negative value in a distribution. The explanation is simple: the three *Dvalue/(var)* distributions jointly define the full conditional. The first is the base definition, and so each row adds to 1.0 as we expect. The second and third distributions are coded as *modifiers*. In order to preserve the constraint that a row in the full conditional sums to 1.0, a row in a modifier must sum to 0.0, and the sum of all modifiers for any one mass element must be such that the element remains in the range [0, 1] at all times.

As a final example, and one which we will carry through the section on query evaluation, consider the network shown in Figure 1. We will model

[4] I hope the interpretation for these simple distributions is self-evident, given the previous discussion and the formatting provided. If it isn't, just skip this example; not much will be lost. The various costs and values are scaled to keep utility in the range [0, 1.0] as described by Cooper.

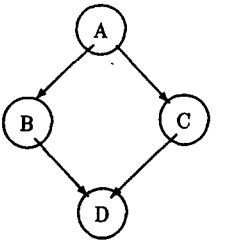

Figure 2. A simple Bayesian network.

both multiple antecedent relationships using a noisy OR structural model. The local expressions for D and E in this network are as follows:

$$exp(D) = 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}},$$

$$exp(E) = 1_{E_t} - c'_{E_t|B_{t,f}} * c'_{E_t|C_{t,f}} + c'_{E_f|B_{t,f}} * c'_{E_f|C_{t,f}}.$$

3. OVERVIEW OF SPI

In this section we briefly review the essential aspects of the SPI approach to inference in Bayesian nets. This will be needed later when we show how the method can be extended to perform inference over nets defined using our local expression language. For further details, see [5] or [20].

Computation of probabilities in a Bayesian net can be done quite straightforwardly, albeit somewhat inefficiently.[5] I illustrate this process with a simple network, shown in Figure 2. First, the marginal and conditional distributions associated with the graph:

$$exp(A) = P_{A_{t,f}},$$

$$exp(B) = P_{B_{t,f}|A_{t,f}},$$

$$exp(C) = P_{C_{t,f}|A_{t,f}},$$

$$exp(D) = P_{D_{t,f}|B_{t,f}C_{t,f}}.$$

[5] We ignore evidence for purposes of this introduction. It introduces only minor complications; see [5, 20] for details.

Now suppose we wish to compute $P(D)$. The procedure is quite simple. We gather all distributions relevant to $P(D)$ (using d-separation, for example), compose them, and sum over all variables except D:

$$P(D) = \sum_{A,B,C} P_{D_{t,f}|B_{t,f},C_{t,f}} * P_{C_{t,f}|A_{t,f}} * P_{B_{t,f}|A_{t,f}} * P_{A_{t,f}}.$$

The actual computation can be optimized somewhat by retaining each dimension only until we have combined with all terms in which the dimension appears (unless that dimension is a goal of the evaluation, in which case it must be retained throughout the computation). For example, we can sum over A, since it is not needed in the final result, immediately after combining $P_{A_{t,f}}$ with $P_{B_{t,f}|A_{t,f}}$ and $P_{C_{t,f}|A_{t,f}}$. Conjunctive queries are handled automatically (we compute the set of relevant variables with respect to the set of variables in the query), and the existence of evidence simply means we must include more distributions in our computation (and normalize).[6] SPI essentially follows this process, but can be viewed as a heuristic procedure for developing factorings which minimize the dimension of intermediate results. The factoring is developed incrementally, and factoring is intermixed with expression evaluation. In the next section we briefly review our current factoring heuristic. A full discussion of the current heuristics for constructing factorings, their theoretical basis, and experimental evaluation of their efficacy appears in a companion article recently published [15].

3.1. The Set Factoring Algorithm

We have developed an efficient heuristic algorithm, called set factoring, for finding good factorings for probability computation. We review it here because we will later extend it for networks in which variable dependence is described using arbitrary local expressions. In the following brief review we will treat each distribution as a subset of the n variables in the network. We will refer to each such subset as a *factor*, and use the following algorithm to combine these factors. Note also that intermediate results will not, in general, be true probability distributions, but will rather be generalized distributions, as defined in [20].

1. Partition the set of factors into independent (i.e., no shared variables) subsets if possible. Evaluate each subset separately using the following method; then combine the results.

[6] There is also a separate phase on arrival of evidence where we sweep all references to unobserved values of the evidence variable from all distributions, but we promised we would not discuss evidence.

2. Construct a *factor set A* which contains all factors to be chosen for the next combination (initially all the relevant net distributions). Each factor in A is represented as a set of variables. Initialize a *combination candidate set B* empty.
3. Add all pairwise combinations of factors of the factor set A to B which are not already in B and in which one factor of the pair contains a variable which is a child or parent of a variable in the second factor, and compute $u = (x \cup y)$ and sum(u) of each pair, where x and y are factors in the set A, and sum(u) is the number of variables in u which can be summed over when the conformal product corresponding to combining the two factors is carried out. A variable can be summed out when it appears in neither the set of target variables nor any of the factors not in the current pair.
4. Choose elements from set B such that $C = \{u \mid u : minimum_B[|u| - sum(u)]\}$; here $|u|$ is the size of u excluding observed variables. If $|C| = 1$, then x and y are the factors for the next combination; otherwise, choose elements from C such that $D = \{u|u : maximum_C(|x| + |y|), x, y \in u\}$. If $|D| = 1$, then x and y are the factors for the next multiplication; otherwise, choose any one of D.
5. Generate a new factor by combining the pair chosen in the above steps. Modify the factor set A by deleting the two factors in the chosen pair from the factor set and adding the new factor in the set.
6. Delete any pair in B which has nonempty intersection with the candidate pair.
7. Repeat steps 3 to 6 until only one element is left in the factor set A, which is the final result.

Following is an example to illustrate the algorithm using the network shown in Figure 2. Suppose that we want to compute the query $p(D)$ for the Bayesian network, and assume that there are two possible values of each variable. The variables relevant to the query are $\{A, B, C, D\}$. We use the set-factoring algorithm to combine the distributions. We will omit domain subscripts on variables for simplicity.

1. Loop1:
   (a) Step 2: The factor set A is $\{P_A, P_{B|A}, P_{C|A}, P_{D|B,C}\}$.
   (b) Step 3: The set B is $\{(P_A, P_{B|A})(P_A, P_{C|A})(P_A, P_{D|B,C})(P_{B|A}, P_{C|A})(P_{B|A}, P_{D|B,C})(P_{C|A}, P_{D|B,C})\}$
   (c) Step 4: The current combination is $(P_A, P_{B|A})$ (there was more than one candidate in this step; we chose one arbitrarily). The result is $P_{A,B}$.
   (d) Step 5: The set A is $\{P_{A,B}, P_{C|A}, P_{D|B,C}\}$.
   (e) Step 6: The set B is $\{(P_{C|A}, P_{D|B,C})\}$.
2. Loop2:
   (a) The factor set A is $\{P_{A,B}, P_{C|A}, P_{D|B,C}\}$.

(b) Step 3: The set B is $\{(P_{A,b}, P_{C|A})(P_{A,B}, P_{D|B,C})P_{C|A}, P_{D|B,C}\}$.
(c) Step 4: The current combination is $(P_{A,B}, P_{C|A})$. The result is $P_{B,C}$.
(d) Step 5: The set A is $\{P_{B,C}, P_{D|B,C}\}$.
(e) Step 6: The set B is empty.

3. Loop 3:
(a) Step 2: The factor set A is $\{P_{B,C}, P_{D|B,C}\}$.
(b) Step 3: The set B is $\{(P_{B,C}, P_{D|B,C})\}$.
(c) Step 4: The current combination in $(P_{B,C}, P_{D|B,C})$. The result is P_D.
(d) Step 5: $\{P_D\}$.
(e) Step 6: The set B is empty.

The factoring result is

$$P(D) = \sum_{B,C}\left[P(D|B,C)\left(\sum_{A}\{P(C|A)[P(B|A)P(A)]\}\right)\right].$$

4. INFERENCE WITH LOCAL EXPRESSIONS

Factoring heuristics for general networks containing arbitrary local expressions are an area of current research in our group. In this section we present a simple extension of the above heuristic which works well for BN2O networks (bipartite graphs with a noisy OR interaction model). This same heuristic has also performed well on a few general multilevel networks, where local expression are a mixture of noisy ORs, additive decompositions, and asymmetric expressions, and has worked well on a few general multilevel noisy OR networks. After a brief presentation of the basic algorithm, we will discuss its operation and present some experimental results.

The algorithm presented above is designed to factor expressions which include only the conformal product and marginalization operators, and takes an input with a single marginalization operator at the outside of the expression. The extended algorithm will have to handle more general expressions, but these will still be of a restricted form. Expressions formed by gathering all variable expressions relevant to a query will still be a marginalization of a conformal product, but the terms of the conformal product will now be general expressions rather than marginal or conditional distributions. The key new decision, then, is when and what to distribute over a + or − factor in the overall conformal product. We integrate this decision into the factoring heuristic described above by expanded the set of candidates generated, revising the interpretation of a

candidate, and extending the scoring heuristic. Each of these is detailed below.

4.1. Extended Candidate Generation

In addition to the pairwise candidates described earlier, we also generate the following combination candidates:

1. The combination of an expression e_1 with *all* expressions containing conditioning variables in e_1 as conditioned variables (e.g., at the first stage of combination with parents). If e_1 is a distribution, then the candidate is generated only if it does not contain all expressions in the factor set A.
2. The transitive closure of the above, that is, the combination of an expression e_1 with all expressions containing as conditioned variables either conditioning variables of e_1 or their parents. If e_1 is a distribution, then this candidate is generated only if it does not contain all expressions in the factor set A.
3. If a factor e_1 is a $+$ or $-$ expression, the combination of e_1 with all remaining factors (i.e., all but e_1).

The basic factoring algorithm is a greedy algorithm. The inclusion of groups of expressions as candidates circumvents some of the limitations of greedy methods. It can be thought of as adding a layer of abstraction to the factoring construction process, although note that "higher level" (i.e., group) candidates must compete not only with each other, but also with simple pairwise candidates.

4.2. Extended Candidate Interpretation

Any pairwise candidate containing a $+$ or $-$ factor as its first factor, as well as any candidate generated by the three extensions above, is interpreted as calling for the application of distributivity, that is, the distribution of the remaining factors in the combination over the first factor. When the selected candidate contains more than two factors, the factoring heuristic is recursively called on the candidate factor set. When a candidate calling for application of distributivity is selected, all factors in the candidate other than the designated $+$ or $-$ factor are distributed over the designated factor, the factoring heuristic is called recursively on the resulting factors, and the results are combined, numerically when possible.[7] That is,

$$Eval((t_{11} + t_{12}) * t_2 * t_3) = Eval(Eval(t_{11} * t_2 * t_3) + Eval(t_{12} * t_2 * t_3)).$$

[7] We evaluate numerically when possible, but in some cases the result of a query will be an expression rather than a distribution.

4.3. Candidate Selection

The heuristic scoring function described earlier is extended as follows:

1. If a candidate C begins with $\pm$ expression, and there exists at least one expression x in set A and not in C for which $T(C) \cap C(x) \neq \Phi$ and $T(C) \setminus C(x) \neq \Phi$, add the number of $\pm$ factors in C to its first stage heuristic value [remember that the first stage heuristic value is just $|T(C)|$, and that lower is better]. $T(C)$ is the set of variables to be retained in the result of evaluating candidate C, and $C(C)$ is the set of conditioning variables in the result.
2. When choosing among candidates with equal first stage scores:
   (a) If both candidates start with distribution-type factors, use the original scoring rule.
   (b) If one candidate starts with a distribution-type expression, and the other starts with a $\pm$ expression, select the candidate starting with the distribution-type expression.
   (c) If both candidates start with a $\pm$ expression, select the one with fewer expressions. If both have the same number of expressions, select the one with the larger number of conditioning variables in the first expression.

The first criterion above biases against using distributivity, because in our current implementation symbolic operations are considerably slower than purely numeric ones. The second continues this bias toward combining distributions first, and adds a heuristic for selecting which $\pm$ expression to distribute over first. The last heuristic in part 2 above is important in BN2O networks. As shown in the experimental results presented later, there is considerable structure which can be exploited, even in networks as richly connected as the QMR-DT basic disease network (4000 findings, 600 diseases, 40,000 disease-finding links). The order in which finding expressions are distributed over is crucial to the successful exploitation of structure.

4.4. Numerical Evaluation of Operations on Distributions

We have already discussed the semantics of overall expressions. In general evaluation of an entire expression for each possible set of variable values would be quite inefficient. Rather, we use the following equivalent method of evaluating expressions:

$*$: *Conformal product.* We use the same procedure as for standard SPI. When combining distributions defined over differing subspaces of the domain for a variable, only those values in the domain for which both distributions are defined need be considered. That is, distributions

are implicitly extended with 0.0 in all values for which they are not defined for variables over which they are defined, and replicated for all values of any variable over which they are not defined. Thus, 1_{D_T} can be seen to specify the distribution {1.0, 0.0} over $\{D = t, D = f\}$.

$\pm$: *Sum/difference.* As before, distributions are extended with zeros for values in the domain over which they are not defined. However, for addition and subtraction we must include domain values over which *either* of the arguments is defined.

5. EVALUATION

5.1. Complexity

5.1.1. FACTORING COMPLEXITY Complexity of the base factoring algorithm is $O(n^3)$ in variables in the network. Since we are only increasing the number of candidates by a constant, 2, complexity is not affected by that change. Similarly, making the factoring algorithm recursive does not change the complexity, since the number of factors is still reduced by one each time through the basic loop. For further analysis see [15].

5.1.2. EVALUATION COMPLEXITY The algorithm reproduces the essential results of Quickscore when applied to two-level bipartite (BN2O) graphs: numerical equation is linear in the number of antecedents, linear in the number of negative findings, and exponential in the number of positive findings. However, further analysis has revealed that is a worst-case result. Often, after distributing over a few expressions, the result can be partitioned into independent subexpressions. In the following we present some preliminary results obtained by applying the above algorithm to the QMR DT network.

5.2 Inference in Bipartite Graphs: QMR DT

QMR DT, at the time of this report, is a BN2O network with over 900 diseases, 4000 findings, and 40,000 disease-to-finding arcs. The size of the network, together with the number of findings in a typical case, makes inference a daunting task. The disease-disease interaction is modeled using a noisy OR, which reduces the inference complexity to linear in the number of negative findings, linear in the number of diseases, and exponential in the number of positive findings (Quickscore [9]). However, a typical case can have up to 50 positive findings, and findings have an average of 10 antecedents (some have hundreds, and these tend to be common findings).

As a result, Heckerman's original trials of quickscore resulted in inference times of 1 min for nine positive findings [9].

5.2.1. LOCAL EXPRESSION REPRESENTATION OF NOISY OR Remember that the local expression language representation we use for the noisy OR is as follows. Given three variables, A, B, and D, with A and B "causes" for D, the expression for $P(D|A, B)$ is

$$exp(D) = 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}}.$$

SPI rewrites expressions for finding variables and immediate successors, eliminating all references to unobserved values of the finding variable. As a result, the expression for a positive finding is reduced to

$$exp(D) = 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}},$$

And that for a negative finding to

$$exp(D) = c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}}.$$

5.2.2. EXPERIMENTS We ran a series of experiments to evaluate the current SPI heuristics on QMR. We ignored negative findings, since they can be handled in linear time and distorted our measurements for small numbers of positive findings. We also ordered positive findings by the number of possible causes (fewer causes first). We did this because findings with fewer causes should be easier to handle (more opportunities for partitioning) and should also be more diagnostic. We then tested each case by querying for the posterior probability of the disease provided with the case, given the first positive finding, the first two positive findings, the first three, and so on, until a threshold of 2,000,000 multiplies was exceeded. (We used a limit of 2,000,000 for an individual query, and the count was reset to zero as each new piece of evidence was added.) as shown in the tables which follow, we then recorded the number of findings, the number of multiplications needed (shown in thousands), and the posterior probability. We chose a limit of 2,000,000 multiples because this corresponded to about 5 min of compute time on a Macintosh Quadra 700 in Common Lisp. We estimate that a straightforward C or Pascal implementation would be about 20 times as fast, and so this is roughly comparable to the 1 min of Pascal computation described in the Quickscore paper. The results in Section 5.2.3 below were obtained by running the test cases in the Sci Am test-case file. Table 1 shows the results of running Quickscore on the ten cases, and in consistent with the results obtained by Heckerman. Table 2 shows the results obtained using the algorithm presented earlier.[8]

[8] For an in-depth study of this particular problem see [7].

Table 1. Sci Am Using Quickscore

Case	Pos. findings	Mults.	$P(D^*)$
1	13	1518	.076
2	13	1262	.999
3	13	596	.282
4	9 (done)	289	.285
5	8 (done)	167	.992
6	12	1617	.136
7	12	1814	.054
8	9 (done)	343	.707
9	12	727	.004
10	8 (done)	191	.875

5.2.3. RESULTS The results are shown in Tables 1 and 2.

Table 2 seems to indicate that the method described here is able to find and exploit considerable structure in the QMR net. Several questions are raised by these results, however. A basic question is one of credit assignment. To what extent is the improvement the result of partitioning versus the distribution heuristic? Table 3 separates these effects by retaining the partitioning step, but choosing randomly which finding to distribute over at each step. As we see, partitioning following each distribution, but randomly choosing the finding to distribute over, typically only increases the capacity by one more finding, and doesn't result in processing of all findings for any additional cases. Adding the distribution heuristic increases the number of positive findings substantially for all cases except 6 and 7, and results in handling all findings for all cases except 1, 6, and 7 within the multiplication limit. For case 1, SPI handles an additional five

Table 2. Sci Am Using Extended Set Factoring

Case	Pos. findings	Mults.	$P(D^*)$
1	18	1571	.916
2	16 (done)	148	1.000
3	15 (done)	321	.330
4	9 (done)	11	.285
5	8 (done)	17	.992
6	15	1527	.111
7	14	1602	.166
8	9 (done)	38	.707
9	17 (done)	1252	.997
10	8 (done)	36	.875

Table 3. Sci Am with Random Choice

Case	Pos. findings	Mults.	$P(D^*)$
1	14	1094	.17
2	14	1455	.999
3	13	506	.282
4	9 (done)	274	.285
5	8 (done)	171	.992
6	12	1045	.136
7	12	1071	.054
8	9 (done)	295	.707
9	14	1999	.279
10	8 (done)	160	.875

positive findings beyond Quickscore. For cases 6 and 7 the picture is less positive: these represent worst-case situations for the method described here. Almost every positive finding has a very large number of antecedents, and so there is little structure to exploit.

6. DISCUSSION

The factoring algorithm we have presented is far from optimal. It is, in fact, a straightforward application of a few basic symbolic algebra techniques to the computation of Bayesian probabilities. We are continuing to refine our formulation of efficient inference using local expressions and at the same time continue our search for effective inference heuristics. What we have attempted to show is two things: (1) that fairly simple algebraic representations can be used to capture structural details not currently recorded in Bayesian nets, and (2) that fairly simple computer algebra techniques can directly and efficiently utilize this structural information. The method we present, implemented as an extension to our factoring algorithm for probabilistic inference, provides a method for performing inference using standard interaction models such as noisy OR within arbitrary Bayesian networks.

Noisy OR is traditionally considered to be of restricted applicability, since standard presentations restrict to the case where all variables take only two values. However, there are generalizations to the multivalued case which require $(d - 1)$ or $(d - 1)^2$ parameters for each antecedent, where d is the number of values a variable can take. For example, the local expression language presented here can represent the noisy-max generalization of Srinivas [22]. Prior work with Intel provides one example of the efficiency of the methods presented. In this work we explored the

application of Bayesian networks to the diagnosis of problems in semiconductor fabrication. One network we constructed to interpret final test results was a BN2O network consisting of 95 (mostly three-valued) variables—64 symptoms and 31 causes—and 155 arcs. With observations on all symptoms, and 27 out of the 64 observations "abnormal," the system processes marginal queries on the cause variables in 5 to 65 s, depending on the variables queried.[9] Profiling tools reveal the computational burden is well balanced: approximately 50% of the CPU time is spent in the symbolic heuristics, and the remainder in numerical computation. This might seem surprising: regardless of whether one measures by number of causes or number of abnormal symptoms, the time complexity might be expected to be on the order to (3^{30}), and therefore the network should be intractable, even using Quickscore. However, this worst case only occurs when every parent is connected to every child, and the Intel network only contains 155 arcs. After a few symbolic distributions of parent distributions over evidence expressions, the query expression is partitionable into several independent subfactors, and so the computation stays tractable. We have had some success in applying the algorithm to multilevel nets as well, although the results are difficult to evaluate, since there are no gold standards against which we can compare.

There are at least two limitations we see with the local expression language proposed here. First, the syntax admits probabilistically incoherent expressions. It would be nice to have a language (like the graphical language of Bayesian nets for large-scale structure) which admitted only probabilistically coherent representations. Second, we see potential utility for a "subset" operator. That is, it is often useful to specify that a probability is constant over some subrange of a conditioning or conditioned variable. For example, the probability of variable B might have one value for a particular value of parent A, and a second value for all other values of A. Currently there is no way to make this structure explicit using our representation.

We began our exploration of probabilistic inference in the context of truth maintenance systems, and at that time used symbolic representation at the level of individual probability mass elements [6]. Later, motivated by efficiency concerns, we changed to a symbolic representation at the distribution level [20]. We now seem to have come full circle: the implementation described here again performs symbolic reasoning on elements as small as individual probabilities. The difference is that we now have a choice of representation grain size, and can select the grain size appropriate for the dependence model being described.

Finally, there is nothing unique to SPI that enables it to be extended in

[9] Times are for Franz Common Lisp on a SPARCstation 1 + .

the ways presented here. Any algorithm for probabilistic inference based on Bayesian nets should be extendable in the ways we have discusses. What is unique is our approach to efficient inference as essentially an algebraic problem, as opposed to a graph-theoretic one. We see no easy way to extend graph-theoretic perspectives, either representationally or inferentially, to include the rich variety of local dependency structures our algebraic language can capture. The view within SPI of inference as essentially a symbolic algebra factoring problem, however, is readily extensible, as we have shown.

7. RELATED WORK

Shachter and Fung [21] have proposed a general representation for contingencies and asymmetries. Their representation is more general than the one described here in one important way: it permits representation of asymmetries which induce cycles. For example, B may depend on C when $A = t$, but C may depend on B when $A = f$. The representation presented here can accommodate such dependency structures, albeit somewhat awkwardly. One must define two contingent versions of B and C, giving them separate names (e.g., $B1$ and $B2$). It is then up to the user to remember to query all instances of B and sum the results.

Geiger and Heckerman [8] have proposed multinets as a general representation for asymmetries. Their representation, however, has a significant limitation: it is not usable as a component in an arbitrary Bayesian net. In particular, they assume that the hypothesis variable of a multinet is a root variable. This restriction does not exist in our representation. However, their representation offers the strong advantage that consistency can be easily established. As we noted earlier, checking the consistency of local expressions in our language is exponential in the number of parent variables. Andreassen and Olesen [2] present a convenient method for describing interaction models such as noisy OR. Dagum and Galper [4] have proposed an additive decomposition of conditional dependence that is easily captured using local expressions. Srinivas [22] has developed a generalized noisy OR model which is compatible in spirit, but might require extension of the set of operators available in our local expression language. This, together with extension to capture influence diagrams, is work in progress.

8. CONCLUSION

Bayesian nets are a compact, intuitive representation for general probabilistic models, but suffer from inability to efficiently represent low-level

structural details such as asymmetries and noisy-OR relationships. We have presented an extension to the belief-net representation for probabilistic models capable of explicitly capturing this information, and shown how the SPI framework can be extended using this information to perform efficient inference. This permits explicit capture of low-level structural details within an arbitrary belief net, and provides efficient processing of arbitrary marginal and conditional queries on the resulting belief net. This facility also provides for easy experimentation on new interaction models, since there is no need to write code to perform inference using the new model: one directly describes the interaction using a simple algebraic local expression language. The full expression language has been implemented and is in use on a variety of monitoring, diagnosis, assessment, and control projects.

ACKNOWLEDGMENTS

Thanks to Bob Fung of Prevision, Inc., and Peter Raulefs and Bob Culley of Intel for many useful discussions. Thanks to NSF (IRI88-21660, IRI91-00530), AFOSR, and Intel for providing the support which made this work possible.

References

1. Agosta, J. M., Conditional inter-causally interdependent node distributions ..., *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 9–16, July 1991.
2. Andreassen, S., and Olesen, K., Specification of models in large expert systems based on causal probabilistic networks, in *Artificial Intelligence in Medicine*, 1993.
3. Cooper, G., A method for using belief networks as influence diagrams, in *Proceedings of the 1988 Workshop on Uncertainty in AI*, 55–63, Assoc. for Uncertainty in AI, Aug. 1988.
4. Dagum, P., and Galper, A., Additive belief-network models, *Ninth Annual Conference on Uncertainty on AI*, (D. Heckerman and A. Mamdani, Eds.), Morgan Kaufmann, 91–98, July 1993.
5. D'Ambrosio, B., Symbolic probabilistic inference, Technical Report, Computer Science Dept., Oregon State Univ., 1989.
6. D'Ambrosio, B., Incremental evaluation and construction of defeasible probabilistic models, *Internat. J. Approx. Reasoning*, July 1990.

7. D'Ambrosio, B., SPI in large BN2O networks, *Tenth Annual Conference on Uncertainty on AI* (Poole and Lopez de Mantaras, Eds.), Morgan Kaufmann, July 1994.

8. Geiger, D., and Heckerman, D., Advances in probabilistic reasoning, *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 118–126, July 1991.

9. Heckerman, D., A tractable inference algorithm for diagnosing multiple diseases, *Proceedings of the Fifth Conference on Uncertainty in AI*, 174–181, Aug. 1989.

10. Heckerman, D., Breese, J., and Horvitz, E., The compilation of decision models, *Proceedings of the Fifth Conference on Uncertainty in AI*, 162–173, Aug. 1989.

11. Heckerman, D. E., *Probabilistic Similarity Networks*, MIT Press, 1991.

12. Henrion, M., Toward efficient probabilistic diagnosis with a very large knowledge-base, *AAAI Workshop on the Principles of Diagnosis*, 1990.

13. Li, Z., A factoring approach to probabilistic inference in belief nets, PhD Thesis, Computer Science Dept., Oregon State Univ., June 1993.

14. Li, Z., and D'Ambrosio, B., An efficient approach to probabilistic inference in belief nets, *Proceedings of the Annual Canadian Artificial Intelligence Conference*, Canad. Assoc. for Artificial Intelligence, May 1992.

15. Li, Z., and D'Ambrosio, B., Efficient inference in Bayes nets as a combinatorial optimization problem, *Internat. J. Approx. Reasoning* 10(5), 1994.

16. Pearl, J., *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, Palo Alto, Calif. 1988.

17. Peng, Y., and Reggia, J., A probabilistic causal model for diagnostic problem solving—part 1: Integrating symbolic causal inference with numeric probabilistic inference, *IEEE Trans. Systems Man Cybernet.* (special issue on diagnosis) SMC-17(2), 146–162, 1987.

18. Raiffa, H., *Decision Analysis*, Addison-Wesley, Reading, Mass., 1968.

19. Shachter, R., Evaluating influence diagrams, *Oper. Res.* 34(6), 871–882, Nov.–Dec. 1986.

20. Shachter, R., D'Ambrosio, B., and DelFavero, B., Symbolic probabilistic inference in belief networks, *Proceedings Eighth National Conference on AI*, AAAI, 126–131, Aug. 1990.

21. Shachter, R., and Fung, R., Contingent influence diagrams, Tech. Report, Advanced Decision Systems, Sept. 1990.

22. Srinivas, S., A generalization of the noisy-or model, *Ninth Annual Conference on Uncertainty on AI*, 208–218, July 1993.